NFV-based Scalable Guaranteed-Bandwidth Multicast Service for Software Defined ISP networks

Hardik Soni, Walid Dabbous, Thierry Turletti, Hitoshi Asaeda

NFV-based Scalable Guaranteed-Bandwidth Multicast Service for Software Defined ISP networks

Hardik Soni, Walid Dabbous, Thierry Turletti, and Hitoshi Asaeda

***Abstract*—New applications where anyone can broadcast high quality video are becoming very popular. ISPs may take the opportunity to propose new high quality multicast services to their clients. Because of its centralized control plane, Software Defined Networking (SDN) enables the deployment of such a service in a flexible and bandwidth-efficient way. But deploying large-scale multicast services on SDN requires smart group membership management and a bandwidth reservation mechanism with QoS guarantees that should neither waste bandwidth nor impact too severely best effort traffic. In this paper, we propose; (1) a scalable multicast group management mechanism based on a Network Function Virtualization (NFV) approach for Software Defined ISP networks to implement and deploy multicast services on the network edge, and (2) the Lazy Load Balancing Multicast (L2BM) routing algorithm for sharing the core network capacity in a friendly way between guaranteed-bandwidth multicast traffic and best-effort traffic and that does not require costly real-time monitoring of link utilization. We have implemented the mechanism and algorithm, and evaluated them both in a simulator and a testbed. In the testbed, we experimented the group management at the edge and L2BM in the core with an Open vSwitch based QoS framework and evaluated the performance of L2BM with an exhaustive set of experiments on various realistic scenarios. The results show that L2BM outperforms other state-of-the art algorithms by being less aggressive with best-effort traffic and accepting about 5-15% more guaranteed-bandwidth multicast join requests.**

***Index Terms*—Load Balancing, Open vSwitch, Multicast Routing, Network Function Virtualization, QoS, Software Defined ISP Networks.**

I. Introduction

The massive increase of live video traffic on the Internet and the advent of Ultra High Definition (UHD) videos have placed a great strain on ISP networks. These networks follow a hierarchical structure for providing Internet access to millions of customers spread over large geographical areas and connected through heterogeneous access technologies and devices. Recently, new over-the-top (OTT) applications, such as Periscope[1] or Facebook Live Stream[2], where anyone can broadcast their own channel are becoming very popular both on smartphones and computers. To support such new streaming applications and satisfy the clients' demands, ISPs may want to enable high-quality multicast services in their networks. One solution is to use built-in multicast within their infrastructure to implement flexible, bandwidth-efficient and scalable multicast delivery services. Such an approach may enable the efficient deployment of many-to-many broadcast services such as Periscope, which could be extended to handle multicast group creation transparently on behalf of users. However, multicast has failed to achieve Internet-wide support so far [1], and even for the limited deployment in managed networks for services such as IPTV, it requires complex integration of specialized multicast enabled routers and protocols, traffic engineering and QoS mechanisms in the networks.

Software Defined Networking (SDN) appears to be an attractive approach to implement and deploy innovative multicast routing algorithms in ISP networks [2], [3], [4], [5], [6], [7], [8], thanks to its logically centralized control plane. More specifically, in Software Defined ISP networks, live video streaming applications can benefit from QoS guaranteed dynamic multicast tree construction algorithms [9], [10], [11], [12], [13] that exploit the global view of the network.

In addition, ISPs could exploit fine-grained control over QoS guaranteed multicast and best-effort traffic to implement traffic engineering policies that are friendly to low priority best-effort traffic. Several advanced multicast routing algorithms integrating load balancing techniques have been proposed in the literature to better utilize the network bandwidth, avoid traffic concentration and limit congestion in the network [5], [6], [9], [10], [11], [12]. However, these approaches require costly real-time monitoring of link utilization to allow network resources sharing between the QoS guaranteed and best effort traffic classes according to ISP traffic management policies.

Moreover, SDN-based centralized architectures suffer from well-known scalability issues. Different approaches based on either distributed [14], [15], [16], [17] and hierarchical [18], [19] control planes or on stateful data planes [20], [21] have been proposed to address SDN scalability issues in general. Distributed controllers usually need costly state synchronization mechanisms. Therefore, only the approaches that propose delegation [18], [22] could be followed but they require to implement the whole functionalities of controllers at each router. Indeed, in the presence of large-scale multicast applications, extra processing is required at edge routers to handle locally all Internet Group Management Protocol (IGMP) membership messages [23] that would otherwise be flooded to the controller.

H. Soni, W. Dabbous and T. Turletti are with Université Côte d'Azur, Inria, France. Email:{ hardik.soni | walid.dabbous | thierry.turletti}@inria.fr

H. Asaeda is with National Institute of Information and Communications Technology, Japan. Email:{ asaeda@nict.go.jp

Corresponding author: H. Soni (email: hardik.soni@inria.fr).

[1]Periscope URL: https://www.periscope.tv/.

[2]Facebook Live Stream URL: https://live.fb.com/

In this work, we address the two following problems: (1) how to avoid implosion of IGMP group membership messages at the SDN controller and (2) how to deploy guaranteed-bandwidth multicast services in Software Defined ISP networks with low cost and while being friendly with best effort traffic.

To address the first problem, we propose to exploit the hierarchical structure of ISP networks and to use Network Function Virtualization (NFV). In short, we delegate when needed the multicast group membership management through specific virtual network functions (VNFs) running at the edge of the network.

To address the second problem, we propose a novel threshold-based load balancing algorithm in which a certain amount of link capacity in the ISP's infrastructure is reserved in priority for guaranteed-bandwidth traffic. This means that in absence of guaranteed-bandwidth traffic, best-effort traffic can use this capacity. Hence, we dynamically increase the capacity share by gradually increasing the threshold. This approach is friendly to best-effort traffic and helps in indirectly load balancing the guaranteed-bandwidth traffic without the need of real-time link traffic monitoring mechanisms, as the controller is responsible for accepting or rejecting multicast subscription requests and is aware of bandwidth requirements and network link capacities.

Our contributions in this paper are the following: (1) an original solution to handle multicast group management in a scalable way on Software Defined ISPs with multicast networking functions running locally on NFV Infrastructure Point of Presences (NFVI-PoPs) and NFV-based Central Offices (NFV-based COs); (2) a smart multicast routing algorithm called L2BM (Lazy Load Balancing Multicast) for large-scale live video streaming applications, which runs on the SDN controller to route the streams across the NFVI-PoPs or NFV-based COs and follows a threshold-based traffic engineering policy for capacity sharing; (3) an implementation of our framework including the scalable group management approach and the L2BM multicast routing algorithm in Open vSwitches (OVS) [24] based QoS Framework using OpenFlow and Floodlight controllers and the evaluation of the L2BM algorithm, with comparisons with state-of-the-art solutions. We provide a web site[3] that includes the implementation of our framework, the simulation code, the scenarios scripts to reproduce all the experiments and extended results obtained with scenarios not shown in the paper.

This paper is an extended version of the work published in [25] in which we originally proposed the L2BM algorithm. In particular, it includes a more complete evaluation of L2BM with a simulator, considering congested traffic scenarios to model flash-crowd and peak hours traffic in Software Defined ISP networks. Moreover, we present in further details the proposed architecture and the implementation of the QoS framework based on Open vSwitches (OVS) [24].

The rest of the paper is organized as follows: Section II presents our architectural approach for deploying scalable, flexible and hierarchical control for multicast group membership management at the network edge, Section III presents L2BM, Section IV describes the implementation of our framework, Section V presents the evaluation of our multicast routing algorithm L2BM, Section VI discusses the related work and Section VII concludes the work.

[3]See https://team.inria.fr/diana/software/l2bm/

II. Scalable Multicast Group Management for Software Defined ISPs

In this section, we tackle the problem of deploying multicast functionalities in a scalable and flexible way on Software Defined ISP networks.

Traditional multicast routing and management protocols such as PIM-SM [26] and IGMP [23] effectively establish and maintain multicast communication paths between sources and receivers. Multicast routers run complex state machine for group membership management of interfaces. In brief, they handle IGMP membership report messages sent by the receivers to manage group membership state, and accordingly send PIM Join/Prune messages to the upstream routers to coordinate the multicast routing paths. Deploying multicast functionalities in SDN without taking precautions can lead to congestion issues at the controller as edge SDN routers need to forward all IGMP because they can not run complex group membership state machines neither store state information to take decisions in an autonomous way.

Let us consider a hierarchical ISP network as shown in Figure 1. With the advent of NFV, network aggregation points at central offices (COs) are being transformed into mini-datacenters. These NFV-based COs gather commercial-off-the-shelf (COTS) hardware that can run any network functions such as NATs, firewalls or caches [27]. Access networks aggregate the customers' access lines at the COs at the frontier of the metro ISP network. The metro ISP network interconnects the COs using relatively high capacity links compared to access network links. Similarly, the core network interconnects gateway Central offices serving as Points of Presence and including NFV infrastructure that we call NFVI-PoPs. With SDN, a controller is responsible for programming packet forwarding in its own domain. In this paper, we refer to it as the Network Controller (NC) of a given domain.

In our approach, NCs delegate multicast group management functionalities to virtual network functions (VNFs) running at NFV Infrastructure at the edge of the metro networks. We call these functions MNFs for Multicast Network Functions, and define MNFs-H and MNFs-N. MNFs-H running in NFV-based COs exchange IGMP query and report messages with the downstream receivers. MNFs-N running in NFVI-PoPs process PIM Join/Prune signals sent by MNFs-H based on the membership states of their NFV-based CO. Unlike traditional PIM Join/Prune messages, these signals do not update trees by themselves, but instead inform the corresponding MNFs-N to coordinate multicast tree update with the NC. Indeed, delegating group membership management processing at NFV-based COs can greatly reduce the concentration of control traffic emerging from multicast applications. Our solution aims to achieve scalability similarly to traditional distributed multicast protocols in SDN architecture in spite of its centralized

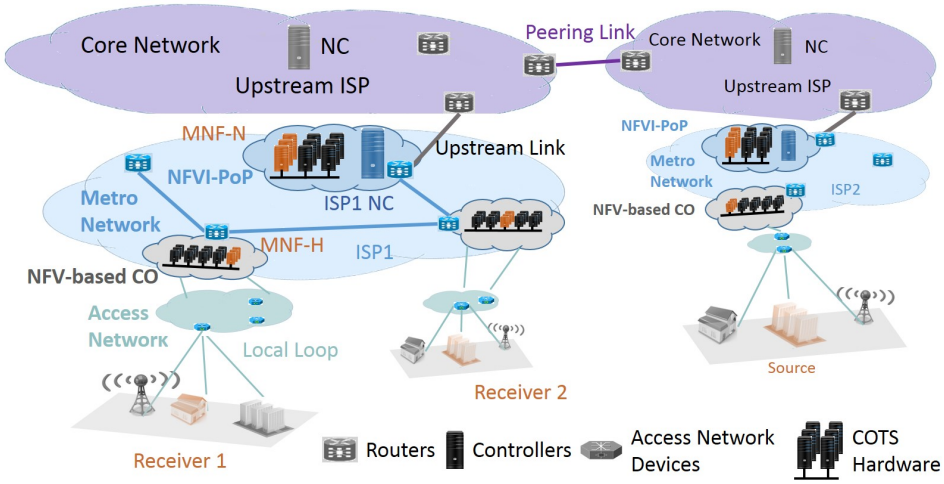


Fig. 1: Example of Software Defined ISP Network with NFVI-PoPs

control plane. It gives flexibility using NFV to enable multicast support on demand and does not put the burden of requiring multicast state management functionality on all the routers and especially core routers. NCs communicate with the NFV orchestrators that run on each NFV-based CO of the domain to instantiate MNFs when necessary. Note that NFV orchestrators are responsible for scaling in/out their VNFs according to the group membership traffic load, providing flexibility. We emphasize that implementing the MNFs functionalities requires several features that are not compatible with hardware SDN routers, which are usually dumb devices. In particular, it is necessary to run a state machine for implementing IGMP and for generating periodically membership queries to the multicast receivers. As described earlier, we argue that the presence of mini-datacenters in central offices (NFV-based COs) as shown in Figure 1 will enable running the MNFs functionalities as VNFs. If such datacenters are not deployed in central offices in the near future, MNFs could either be implemented as middleboxes running next to edge routers or integrated within software routers as switching at the edge is becoming virtual, handled on x86 cores as anticipated by SDNv2[4].

Let us now examine our proposed architecture with an example. At the start, in absence of MNFs running at the access NFV-based COs, the first group join request among the receiver hosts is forwarded as a packet-in to the metro NC. If the corresponding source or the multicast tree is already present in the metro network, then the metro NC establishes[5] the bandwidth guaranteed path for the requested multicast flow between the edge router that receives the join request at the access NFV-based CO and the multicast tree. At the same time, the metro NC interacts with the NFV orchestrator of the access NFV-based CO to instantiate an MNF-H and with the NFV orchestrator co-located at the NFVI-PoP to instantiate an MNF-N. After that, the group specific IGMP traffic received by the edge router is redirected to the MNF-H and handled locally. In addition, the PIM Join/Prune signaling traffic is redirected to the MNF-N that manages group membership of all the NFVI-based COs and communicates with the metro NC to update the multicast tree for each group in the metro network. In case the access NFV-based CO is already receiving the requested multicast flow, the MNF-H is responsible for configuring the edge router to forward the multicast flow to the port where the IGMP membership report has been received. Once the processing of IGMP messages are delegated to MNF-H, both the metro NC and MNF-H can configure the SDN edge routers. This design makes all the flow tables in the edge router vulnerable to unauthorized modification from the corresponding MNF. Hence, careful programming of MNFs is required to avoid race conditions on flow tables and maintain consistency in routers tables.

Metro NCs inform upper level NCs in the hierarchy of the presence of all the multicast sources in their domain and also exchange this information with peering ISPs' NCs. We assume that the streaming application implements the required signaling protocols such as SIP, SAP and MSDP to announce and discover multicast groups.

On detecting a multicast source, an NC communicates with the orchestrator on its local NFVI-PoP to instantiate an MNF-N if the latter is not yet running, in order to store information on the new multicast source and process future Join/Prune signals. If neither the source nor the multicast tree corresponding to the PIM Join signal belongs to the domain, the MNF-N forwards the PIM Join signal to the upstream network through the upstream route set by the NC. If the source and the receivers are not in the same ISP, the Join signal will propagate through the peering link to reach the MNF-N corresponding to the source's ISP and a bandwidth guaranteed path will be established on both ISPs.

MNF-Hs are responsible for aggregating the group membership reports received from their NFV-based CO networks, and according to the state machine, they can send Join/Prune signals to the MNF-N for the different multicast groups.

[4]See article "Time for an SDN Sequel? Scott Shenker Preaches SDN Version 2," www.sdxcentral.com/articles/news/scott-shenker-preaches-revised-sdn-sdnv2/2014/10/.

[5]The algorithm used to dynamically construct multicast trees with bandwidth guarantee is described in Section III.

Hence, similar to multicast routers in traditional multicast protocols, MNFs can maintain the group membership state of their downstream receiver hosts. Figure 2 illustrates our approach.

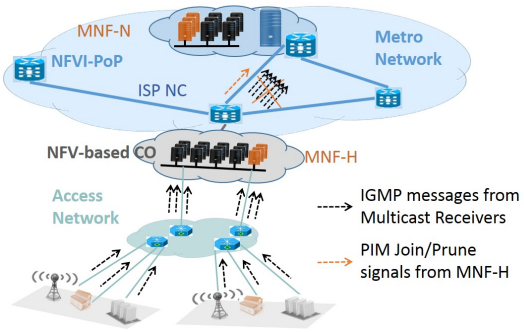


Fig. 2: Multicast Group Membership Management Analysis

Without the deployment of the MNF-H, the edge routers do not maintain multicast state and do not take decision to replicate the multicast stream to the required downstream interfaces in the centralized SDN based approach. Hence, all the multicast group membership messages are forwarded to the NC. In our proposed approach, once the MNF-H at the access NFVI-based CO receives a multicast stream, all the successive IGMP join requests received for the group at the edge router from the downstream access network are locally handled by the MNF-H. Hence, irrespective of the number of multicast group membership messages received for a group from end hosts in an access network, only the first IGMP join and the last IGMP leave requests result in sending a PIM Join/Prune signal from MNF-N to the metro NC in order to add/remove the NFVI-based CO from the multicast tree of the group. Therefore, with this mechanism NC is involved only for routing in core network and does not have to maintain IGMP state machines at any of the end hosts.

In Section IV, we describe the proof-of-concept implementation of MNFs with Open vSwitches that we use to validate our proposal of delegating group membership management to VNFs for scalability.

III. Lazy Load Balancing Multicast routing Algorithm (L2BM)

In this section, we describe L2BM, a threshold-based load balancing routing algorithm proposed to deploy a guaranteed-bandwidth multicast service in ISP networks. L2BM dynamically builds a multicast tree to deliver traffic to member NFVI-PoPs in Core networks or NFVI-based COs in ISP's Metro. It routes multicast streams on-demand to the member NFVI-PoPs or NFVI-based COs in the Core or Metro network, respectively, by programming SDN enabled forwarding devices in the network. L2BM attempts to be friendly with best-effort traffic and remove the need of real-time link measurement mechanisms, which are usually required when deploying load balancing mechanisms.

The main idea is to reserve in priority a certain fraction of link capacity, referred as the *threshold*, for guaranteed-bandwidth multicast services and to restrict the corresponding traffic to this threshold through traffic shaping. Then, to make sure that the best-effort traffic can use the reserved link capacity in the absence of guaranteed-bandwidth traffic, we use in forwarding devices Hierarchical Token Bucket [28], which is a classful queuing discipline that allows sharing the link capacity with flows of different priorities. More precisely, we associate a threshold parameter to each multicast group join request received at NFV-based COs. While connecting the NFV-based CO serving as a node to the multicast tree of the requested group, the L2BM algorithm avoids the links with utilization equal or greater than the current threshold value. L2BM attempts to reserve the required bandwidth on the minimum length reverse path from the receiver to any neighboring node in the multicast tree. If no reverse path to the tree can be found, L2BM increases the threshold value to consider previously avoided links and retries to attach the receiver to the target multicast tree. In presence of multiple shortest paths length with equal threshold value, L2BM selects the one with the least maximum utilization for guaranteed traffic among its links. This information is available at no cost at the NC as it keeps track of previous requests.

Algorithm 1 shows the pseudo-code of L2BM for adding a new node in the multicast tree, using notations defined in Table I.

TABLE I: Notations used in Algorithm 1

Symbols	Definition
$\mathcal{E}$	set of edges
$\mathcal{V}$	set of nodes
$\mathcal{V}_\mathcal{T}$	$\mathcal{V}$ for multicast tree $\mathcal{T}$ of group M
e_{vu}	edge from v to u
B_{vu}	link bandwidth consumption of e_{vu}
C_{vu}	link capacity of e_{vu}
U_{vu}	link utilization of e_{vu}; $U_{vu} = B_{vu}/C_{vu}$
θ	threshold parameter
θ_{init}	Initial value of θ
θ_{max}	Maximum value of θ
r	new receiver for group M
b	bandwidth requirement of group M
$\mathcal{P}$	FIFO queue, stores pruned nodes
$\mathcal{Q}$	FIFO queue, stores nodes to be explored
u	head node of the $\mathcal{P}$ queue
$Path[v]$	set of edges constructing path from v to r
$len(v)$	path length from node v to r
$U_{max}(Path[v])$	Max. link utilization of edges in $Path[v]$

Line 2 initializes the FIFO queue $\mathcal{Q}$ of graph nodes to be explored and path related variables. These graph nodes represent the SDN enabled forwarding devices in the Core or Metro network that compose the multicast tree. Then, the graph search starts with the initial value of the threshold (θ_{init}) to find a reverse-path from the new receiver, r, to the closest node in the multicast tree of the requested group. The threshold, θ_{init}, allows the links to be loaded to a specified value, before starting spreading the multicast traffic by taking longer paths. The algorithm performs Breadth First Search (BFS) to find a path to the tree considering only edges utilized below current threshold θ and pruning the rest. In line 5, the $ThresholdBFS$ function initializes the prune queue $\mathcal{P}$ and sets the local parameter θ_{next} to 1. The θ_{next} variable is used to record the minimum θ value required for the next recursive call of the $ThresholdBFS$ function. Queue $\mathcal{P}$ stores the nodes having any of their edges pruned due to an utilization

higher than the threshold θ. In case none of the tree nodes is found, a recursive call to the function is done with queue $\mathcal{P}$ (the pruned nodes) and θ set to θ_{next} to continue the search. The loop in line 6 starts the graph search from u, the head node of the queue $\mathcal{P}$. If node u is part of the tree for the requested multicast group, the algorithm terminates (lines 8-10). Line 11 adds node u in the *visited* set. The algorithm expands the search by considering each incoming edge e_{vu} to node u (line 12). It computes U^{new}, the new link utilization of the edge e_{vu} by adding the bandwidth demand b (line 13). If U^{new} is higher than the maximum capacity, θ_{max}, allocated for guaranteed traffic it discards the edge (lines 14-16). Otherwise, it further checks U^{new} against the current value of the threshold θ (line 17). If U^{new} is below θ, $Path^{new}$ and len^{new} are computed to reach v via u (lines 18,19). If another path of the same length to v already exists, the algorithm updates $Path[v]$ with the one having the lowest maximum edge utilization of the two (lines 20-23). Otherwise, node v is added in the queue Q and $Path$ and len are updated for v (lines 24-27). If U^{new} is above θ (lines 29-32), node u is added in prune queue $\mathcal{P}$ and θ_{next} is set to the minimum edge utilization value among all the pruned edges. If no tree node is found in the current search, the algorithm removes the nodes stored in $\mathcal{P}$ from *visited* set to consider pruned nodes in the next recursive call (line 36). Then, it makes the recursive call to the function with prune queue $\mathcal{P}$, round up to tenth of θ_{next} and *visited* set as input parameters (lines 35-38).

The algorithm is now explained through a simple example. Figure 3 shows a network with 4 nodes and a sequence of 3 events marked with numbers from T_0 to T_2. In this example, all links have the same capacity of 100 units. The current load and percentage link utilization are shown for each link e_{vu} using the notation (C_{vu}, U_{vu}). T_0 (in black) corresponds to the initial network state with existing multicast traffic. At T_1 (in red), receiver R_1 from node B joins a new multicast group with source S at A, which requires 20 units of bandwidth. As the link utilization of both e_{AB} and e_{DB} is equal to 0.1, the algorithm will explore both the edges, but will select tree node A increasing U_{AB} up to 0.3. At T_2 (in green), receiver R_2 from node D joins the same multicast group with source S. Again L2BM starts with $\theta = 0.1$, but ends up selecting the path A-C-D.

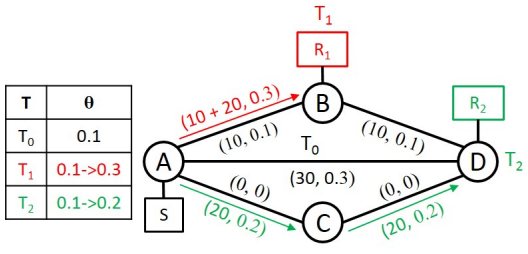


Fig. 3: Example of L2BM functioning

When all the members from an access network leave a particular group, the MNF from the corresponding NFV-based CO has to notify the NC to remove its membership from the multicast tree. Node deletion from the tree is done by recursively removing the non-branch nodes in the reverse path of the stream till a branch node is encountered. This approach does not perturb the existing multicast tree, which prevents packet loss and reordering problems that could have emerged when restructuring the tree.

For each multicast join request, L2BM starts with an initial threshold value of θ_{init}. L2BM performs BFS using only edges with utilization below or equal the threshold. Nodes with higher link utilization are saved in prune queue $\mathcal{P}$ to continue, if needed, the search with increased threshold value through recursive calls. Let us consider the worst case scenario in which each call of *ThresholdBFS* visits only one node

Algorithm 1: Lazy Load Balancing Multicast

```
1  Function AddReceiverInTree(): Path
2      Q.Enqueue(r), len(r) = 0, visited ← ∅
3      return ThresholdBFS(Q, θ_init, visited)
4  Function ThresholdBFS(Q, θ, visited): Path
5      P : To store pruned nodes, θ_next = 1
6      while Q ≠ ∅ do
7          u ← Q.Dequeue()
8          if u ∈ V_T then
9              return Path[u]
10         end if
11         visited ← visited ∪ u
12         foreach e_vu and v not in visited do
13             U^new ← U_vu + b/C_vu
14             if U^new ≥ θ_max then
15                 continue
16             end if
17             if U^new ≤ θ then
18                 Path^new ← Path[u] ∪ e_vu
19                 len^new ← len(u) + 1
20                 if v ∈ Q then
21                     if len(v) = len^new and
                        U_max(Path[v]) > U_max(Path^new)
                        then
22                         Path[v] ← Path^new
23                     end if
24                 else
25                     Q.Enqueue(v)
26                     len(v) ← len^new
27                     Path[v] ← Path^new
28                 end if
29             else
30                 P.Enqueue(u)
31                 θ_next = min(θ_next, U^new)
32             end if
33         end foreach
34     end while
35     if P ≠ ∅ then
36         visited ← visited \ {v : ∀v ∈ P}
37         return ThresholdBFS(P, ⌈θ_next × 10⌉/10,
            visited)
38     end if
39     return NULL
```

and the rest of the nodes are enqueued in $\mathcal{P}$. This leads to at most 10 consecutive calls of $ThresholdBFS$ as the algorithm increases θ and rounds it up to tenth of the minimum of all link utilization operating above current θ, and each edge is visited exactly once. Hence, the order of run time cost of L2BM is the same as the one of BFS, $O(|\mathcal{V}| + |\mathcal{E}|)$.

IV. Testbed and Simulator Evaluation Frameworks

In this section, we first describe the proof-of-concept framework, based on an SDN controller and Open vSwitches, which implements the MNFs with the edge-based group management support and L2BM algorithm for Software Defined ISP networks. Then we present the simulator we implemented to evaluate the performance of L2BM on networks with high link capacity using different traffic scenarios, in a time efficient manner. The performance evaluation mechanisms and results are described in Section V using both setups.

A. Open vSwitch based QoS Framework

Providing guaranteed-bandwidth multicast services in Software Defined ISP networks requires allocating bandwidth resources on the multicast trees' links in a programmatic way. In particular, data plane network devices in the networks have to support QoS-based forwarding on the traffic flows and should be programmable. However, existing SDN protocols such as OpenFlow (OF) [29] have limited support to program QoS on switches. To demonstrate the feasibility of guaranteed-bandwidth routing for multicast services in real networks, we implemented an SDN-controller module that provides the mechanisms to allocate bandwidth resources at the granularity of flow definition. In the following, we describe the implementation choices made for the SDN-controller module.

The OpenFlow protocol has gained widespread acceptance to implement the southbound interface of SDN, even though its specifications and features are still evolving. From the very first version of OpenFlow, programmable QoS support[6] was provided through simple queuing mechanisms, where one or multiple queues are attached to the switch ports and flows are mapped to a specific queue to satisfy QoS requirements. In practice, the OF controller uses the *ofp_queue_get_config_request* message to retrieve queue information on a specific port. Then, OF switches reply back to the controller with *ofp_queue_get_config_reply* messages providing information on the queues mapped to the port. The OpenFlow version 1.3 and later versions specify the *Meter Table* feature to program QoS operations on a per-flow basis. However, traffic policing mechanisms such as rate-limiting throttle the bandwidth consumption of links by dropping packets, which may be problematic especially for time-sensitive flows. So, we rely instead on the Linux kernel QoS features and in particular, on the *tc* Linux command to configure Traffic Control (e.g., shaping, scheduling, policing and dropping) in a way it allows supporting bandwidth guarantee without loss in order to provide high QoE video [30].

[6]Note that queue creation and configuration are outside the scope of the OF protocol.

LINC[31], CPqD[32] and Open vSwitch (OVS) are widely used software switches among existing implementations of OF-based software switches. We chose OVS because it has been shown that it can be ported to hardware [33] and achieve carrier-grade switching capability using specific acceleration software [34]. OVS offers a wide range of protocols like sFlow, NetFlow but it does not implement any QoS mechanism itself. However, it provides the support to configure its OF-enabled switch ports with a subset of Linux kernel QoS features that is sufficient to implement guaranteed-bandwidth multicast routing.

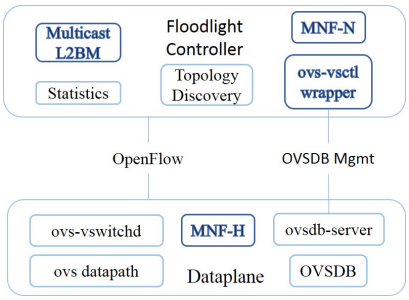


Fig. 4: Queue-based QoS integration with OVS and Floodlight

To summarize, OVS is a multi-process system whose daemon called *ovs-vswitchd* is used to control all the software switch instances running on the hosting machine. *ovs-vswitchd* reads configuration from a database called Open vSwitch Database (OVSDB). While OF is used to program flow entries, the Open OVSDB management protocol is used to configure OVS itself through the *ovs-vsctl* command-line utility. The latter connects to the ovsdb-server process that maintains the OVS configuration database. *ovs-vsctl* is particularly important because it can create/delete/modify bridges, ports, interfaces and configure queues using the Linux kernel QoS features.

In order to provide an API for handling queues on remote software switches, we implemented a wrapper for *ovs-vsctl* and added it as a module to the Floodlight controller. We chose floodlight because of its simple and modular architecture along with ease of use. The controller module exposes APIs for queue management on data plane interfaces of OF forwarding devices, and through these APIs, other applications can perform traffic shaping, policing and scheduling features provided by *linux-tc*. This approach creates dependency on *ovs-vsctl* utility, but it avoids re-implementation of the OVSDB management protocol[35] on the SDN controller. We evaluated our multicast service deployment solution in the Open vSwitch based QoS framework considering a two-level hierarchy ISP network, i.e., with metro and access networks.

MNFs implement IGMP and PIM message handlers. Those handlers require to maintain elaborate multicast state to aggregate membership messages received from downstream networks. MNFs also communicate with the controller to handle groups join/leave when needed and to receive the corresponding multicast streams based on membership status on downstream networks. We implemented MNF-H as a standalone software running within the same operating system environment as Open vSwitches to process locally IGMP

messages. On the other hand, MNF-N is implemented as an independent server module in the Floodlight controller. The MNF-N server communicates with MNF-H to receive PIM Join/Prune signals but does not use the OpenFlow channel. We took this implementation decision to align it with the approach described in Fig. 1. The major functions of MNFs-H are to maintain multicast state, aggregate the IGMP messages received from the switch interfaces and update the flow tables in the switch accordingly. MNFs-H handle group membership management for IGMP messages received from end receivers. This requires updating the flow table in the switch programmatically. Hence, we use the lightweight OpenFlow driver *libfluid*[36] to program the Open vSwitch and update the multicast group table as required. Such a local processing of group membership messages using network functions prevents the repetitive forwarding of group membership messages from the same edge nodes to the controller.

B. The Simulator

We have implemented a simulator that is able to generate different traffic scenarios, execute extensive experimental runs in time efficient manner and store a snapshot of network link utilization metrics after each multicast join and leave event in the network. Concerning link capacity allocation, the OVS-based L2BM testbed explicitly allocates link capacities to provide bandwidth guarantee to the multicast streams without the need of costly link monitoring system. To do that, it maintains the graph data structure of the topology, updates the links' consumption and allocates the bandwidth using the OVS-wrapper. Our simulator has exactly the same behavior without the need for emulating the network and making OVS-wrapper calls to allocate link capacity. More precisely, it simulates the Internet2-AL2S network topology by maintaining a graph data-structure and update the bandwidth consumption of each link based on routing decisions made by the multicast algorithms for each event. In this way, the most recent network view is provided to the running algorithm after each event.

However, in the testbed, multicast events may be processed in a different order than their arrival order. This mismatch is due to the controller implementation (e.g., single or multi-thread) and to delay variation in packet_in events sent to the controller by the switches. The simulator generates the multicast events trace with time stamped events according to traffic scenarios and sequentializes the events with nanosecond precision before feeding the event trace to the different algorithms. It does not wait between two events, thereby accelerating the execution of experiment run.

Unlike the testbed, the simulator does not run any SDN controller. Instead, it converts multicast join and leave events into function calls in the implementation to add or remove the receiver in the target multicast group. Hence, in the simulator, all the algorithms receive each event in the same order for a given experimental run even if the events are generated with very small time difference. However, in the case of the testbed, the events received in time duration of order of microseconds are simultaneously received by the controller without guaranteeing a specific processing order in the same experimental run across all the algorithms.

To compare the performance results between the testbed and the simulator, we executed 20 runs of the different workload experiments in the testbed environment, recorded these 20 event traces and fed them to the simulator. The results of Figure 5a with 95% confidence intervals show the same behavior of the routing algorithms in the simulator and in the testbed. For more details, see Section V-E.

V. Evaluation of L2BM

In this section, we study the performance of L2BM for routing guaranteed-bandwidth multicast flows in a single domain ISP. The evaluation is done using the testbed and the simulator that are both described in Section IV. Without loss of generality, we consider that the guaranteed-bandwidth traffic is allowed to use the whole link capacity of the network (i.e., $\theta_{max} = 1$). We also assume that routers implement the Hierarchical Token Bucket queuing discipline so that best effort traffic can use the reserved bandwidth in the absence of guaranteed-bandwidth traffic.

A. Alternative Algorithms for L2BM

We compare L2BM with two multicast routing algorithms that implement greedy heuristics of the Dynamic Steiner Tree (DST) [37] algorithm with two different metrics: path-length (DST-PL) and link utilization (DST-LU). L2BM allocates the required bandwidth along the path to the new receiver. If it cannot find a path with enough available bandwidth for the multicast group requested by the new receiver, then it rejects the join request. We also implemented the guaranteed-bandwidth and admission control features in DST-PL and DST-LU algorithms. Note that DST-PL with guaranteed-bandwidth and admission control is an alternative implementation of NNFDAR presented in [13]. Both L2BM and DST-PL follow a *nearest node* approach with the path length metric proposed in [37], but in addition, L2BM attempts to limit the maximum link utilization below some threshold. With DST-LU, new receivers join the existing multicast tree using the path with the minimum total link utilization. As the initial threshold (θ_{init}) controls the multicast traffic load allowed on links before triggering load balancing, we use L2BM with low (0.1, 0.2), medium (0.4) and high (0.6) initial thresholds, θ_{init}, to study load balancing for different initial traffic loads on links.

B. Testbed and ISP Network Topology

Testing our bandwidth allocation implementation with Open vSwitches and Floodlight requires a testbed capable of emulating QoS-oriented SDN experiments. Existing emulation tools like Mininet [38] do not consider physical resource constraints, hence the experiments can suffer from errors emerging from resource limitations of the host physical machines. We used the DiG [39] tool to automate the procedure of building target network topologies while respecting the physical resources constraints available on the testbed. Regarding the network topology, we chose *INTERNET2-AL2S* [40] to represent an ISP network with 39 nodes and 51 bidirectional edges. Then

we virtualized this topology using DiG on the Grid'5000 large-scale testbed. DiG implements routers using Open vSwitches and we configured it to allocate sufficient processing power (i.e., two computing cores) at each router for seamless packet switching at line rate. As the grid network uses 1Gbps links and *INTERNET2-AL2S* operates with 100Gbps links, we had to scale down the link capacity to 100Mbps in our testbed experiments. But, we use the simulator for the same network with 1, 10 and 100Gbps link capacities that are representative of links in different tier ISP networks. We present results with network of 1Gbps links and other results made are available[7].

C. Multicast Traffic Scenarios

Properties of the multicast traffic vary according to the level we are in the hierarchy of ISP networks. The controller of a lower level ISP backbone network may be exposed to high end-viewers churn for a multicast session. Indeed, as each NFV-based CO covers a smaller number of end-receivers, their group memberships can be more volatile than for NFVI-PoPs of higher level ISP networks, which have a higher probability of maintaining their group memberships. To account for that in our evaluation, we generate workloads *without churn* and workloads *with churn* to simulate traffic conditions at a high and low level, respectively, in the hierarchy of ISPs networks. At a high level in the hierarchy, each NFVI-PoP aggregates a large number of multicast end receivers, resulting in static multicast sessions. In this case, NFVI-PoPs dynamically join multicast groups, but do not leave their groups, so such a traffic consumes the reserved network capacity θ_{max}, rapidly. Conversely, at a low level, end-receivers in the network join and leave multicast groups during the runs and NFV-based COs also change group membership, frequently.

In all the simulations, we generate different multicast traffic workloads by varying the number of multicast groups from 10 to 150 for a network with 1 Gbps link capacities to load the links sufficiently. We uniformly select multicast group session bandwidth demands for each multicast group in the workload from 1, 2, 5, 9, 12, 25, 50, 80 Mbps representative of different qualities of live video streams (SD, HD, UHD). Concerning the testbed, as we used 100 Mbps link capacity due to physical resource constraints, we generate workloads with lower bandwidth demands for multicast groups uniformly selected from 2, 5 and 8 Mbps.

To generate the workloads without churn, we launch multicast group senders with inter-arrival time exponentially distributed with a mean of $3s$ in order to avoid flash crowd and still have reasonably short simulation time. Once the sender of a multicast group appears in the system, join requests from receivers are sent with an inter-instantiation time exponentially distributed with a realistic mean $1/\lambda$ set to $5s$ [41]. We generate the workloads with churn in a similar way but additionally enforce receiver nodes to leave the multicast group sessions at exponentially distributed time with a mean $1/\mu$ empirically set to $75s$. With this setup, each multicast group session in workload with churn has an average number of $(\lambda/\mu = 15)$ receivers at steady state. We chose to have a moderate number of users in order to avoid full broadcast scenarios, in which all routers in the metro network would belong to the multicast tree. We simulate the churn of receiver nodes in each multicast group session for a duration of $300s$. Note that we do consider source popularity in this study. This could be an interesting future work using our publicly available source code.

Furthermore, as proposed in [42], we consider two types of multicast traffic: *Homogeneous* and *Concentrated*. Workloads with Homogeneous multicast traffic (called *Homogeneous workloads*) generate multicast flows by utilizing the whole network in equal proportion. Such workloads are useful to compare the results obtained from the testbed and the ad-hoc simulator without limiting the experiments to a specific scenario. By contrast, workloads with Concentrated multicast traffic (called *Concentrated workloads*) aim to capture the congestion and traffic concentration on critical links for real-world scenarios, e.g., increase in video traffic during peak hours, live transmission of local events, etc. This usually results in higher load in some parts of the network.

We use Homogeneous workloads without churn to validate the results obtained from the ad-hoc simulator against the ones from the testbed. To this end, we distribute receiver and source nodes uniformly across the network and select traffic demands of multicast groups with a uniform probability from 2, 5 and 8 Mbps. To generate the Homogeneous workloads, we distribute the location of sender and receivers of each multicast group in a uniform way across the nodes in the network and we associate 13 receivers to each multicast group. Then we make 20 workload runs to compare the results obtained from the testbed and from the simulator as shown in Section V-E1.

Concentrated workloads are used to emulate realistic traffic concentration scenarios while analyzing the performance of the different load-balancing algorithms with the simulator. To generate them, we use a non-uniform probability distribution to select sources and receivers among network nodes. More precisely, we randomly select a node, representing a hotspot, with a uniform distribution for each workload run. Then we compute the shortest path lengths from this node to all the other nodes in the network. We use the negative of these path lengths as exponents to compute the exponentially scaled distances from the node to all the other nodes. After that, we divide the exponentially scaled distance of each node with the sum of the distances of all the nodes to obtain the probability distribution of the nodes. To generate the Concentrated workloads without churn, we enforce congestion in hotspots with traffic generated by a subset of nodes in the vicinity. More precisely, we select 13 out of 39 nodes as receivers in the network and a source with the above probability distribution of the nodes for each multicast group. This set up has been chosen empirically to create congestion only on small parts of the network, not in the whole network. Finally, to generate Concentrated workloads with churn, we select the source nodes using only 33% of total network nodes. Then we instantiate 20 receivers for each multicast group, similar to multicast group sessions without churn and use an $M/M/\infty$ queue for each group to generate churn for a duration of $300s$. We execute 500 simulation runs of each workload for each algorithm to allow fair comparison among DST-PL,

[7]See URL https://team.inria.fr/diana/software/l2bm/

DST-LU and L2BM routing algorithms.

D. Evaluation Metrics

Measure of Link Utilization: We compute the three following measures of link utilization to evaluate the performance of the different multicast routing algorithms with the workloads described above: 1) Average (Avg) refers to the average utilization of all the links in the network, 2) Standard Deviation (StdDev) estimates the imbalance of traffic spread across the links and 3) Critical Links denotes the percentage of links with utilization higher than 90%. These three different measures of link utilization are used to analyze the network bandwidth consumption and qualitatively estimate the impact of guaranteed-bandwidth flows on best-effort traffic for the different algorithms. A high Avg value means that best-effort traffic will have less overall network bandwidth available. The StdDev measure illustrates uneven spread of available bandwidth across the network links. In particular, a high value of StdDev means a high probability of congestion for best-effort flows and unfair share of link capacities across the network between best-effort and guaranteed-bandwidth traffic. Finally, the Critical Links measure is used to estimate the concentration of guaranteed-bandwidth traffic in the network.

In the case of scenarios without churn, we use a snapshot of network links' utilization once all receivers have joined their multicast groups. Then we compute the average of the metrics over all the runs of the workload with the same number of multicast groups. We use the workloads by varying the number of multicast groups from 10 to 150 to study the behavior of the algorithms when increasing the traffic.

Regarding scenarios with churn, we take a snapshot of network links' utilization at every second, compute the Avg, StdDev and Critical Links metrics along with the Exponential Moving Average of the metrics to study the behavior of the different algorithms over the run duration. We validate the $M/M/\infty$ queue based simulation model as described in Section V-B by studying the link metrics over the entire period of a single workload run involving 130 multicast groups. For all the metrics, we compute an exponential moving average with a smoothing factor of 0.6 for the whole run duration. Apart from validating the simulation model, we study the performance of all the algorithms over multiple runs similar to the scenarios without churn. Also, instead of taking a single snapshot of network links' utilization, we compute the average over the entire run of experiments.

Apart from link utilization metrics, we use the *Bandwidth Demands Acceptance Ratio* to compare the performance of the different algorithms. This ratio is obtained by computing the number of successful join group requests over the total number of join group requests sent. We compute this metric for scenarios with and without churn and for each run of the workload experiments. Then we average it over multiple runs and we plot the 95% confidence interval.

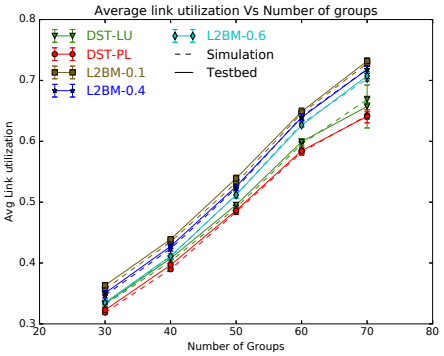
E. Results and Analysis

First, we study in Section V-E1 the link evaluation metrics obtained by running experiments on the testbed and on the ad-hoc simulator using the same workloads. After this analysis, all the results shown in the rest of the paper are obtained using the ad-hoc simulator and with the Concentrated workloads. Second, we study the performance of the different algorithms with the Concentrated workloads without churn in V-E2. Third, we validate the $M/M/\infty$ queue model for the Concentrated workloads with churn in V-E3 then we analyze the performance results obtained with the Concentrated workloads with churn in V-E4.

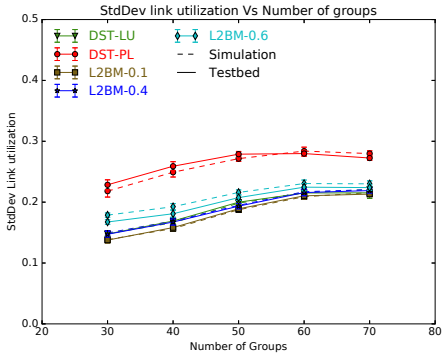
1) Validation of the Simulator with the Testbed: In order to compare performance results obtained with the testbed and the simulator, we perform 20 runs of each Homogeneous workloads without churn and plot average for each metric with 95% confidence interval in 5. Figures 5a-d show superimposed Avg and StdDev measures of link utilization along with the Critical Links metric obtained with the simulator and the testbed experiments for DST-PL, DST-LU and L2BM and for three different values of the initial threshold $\theta_{init} = \{0.1, 0.4, 0.6\}$.

As we can observe, the performance obtained with the simulator for each algorithm closely follows the results obtained in the testbed environment, which validates the implementation of the simulator. From now on, we use the simulator along with the Concentrated workloads to further study the performance of the different algorithms.

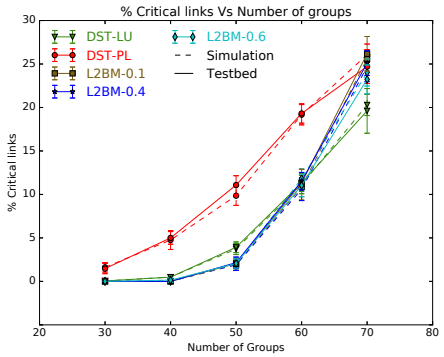
2) Simulation Results using Workloads without Churn: Here we evaluate the algorithms using multicast group session without churn of receiver nodes with the Concentrated workloads described in V-B. Figures 6a-d show the Avg and StdDev measures of links' utilization, along with the Critical Links and bandwidth demands acceptance ratio metrics obtained without churn. DST-PL obtains 5-15% lower Avg, 5% higher StdDev and 5-12% lower bandwidth demands acceptance ratio compared to L2BM with $\theta_{init} = 0.1$. Indeed, DST-PL computes the shortest path to the multicast tree, which rapidly congest the critical links around the hot spots of network. For low traffic workloads (i.e., 10 to 60 multicast groups), all the algorithms accept 100% of the bandwidth demands. We recall that for all the algorithms, multicast join requests are satisfied only if sufficient bandwidth can be reserved on the links. The three variants of L2BM (particularly, $\theta_{init} = 0.10$) have 5% higher Avg link utilization than DST-LU and DST-PL. But unlike the rest of the algorithms, at low traffic the L2BM algorithms obtain zero Critical Links because it aggressively minimizes the maximum utilization of the links by not using the links operating above current threshold, see Figure 6c. We can note that the StdDev for DST-LU and L2BM does not vary significantly, so the two algorithms equally distribute the traffic across the network. However, we can observe for moderate traffic workloads (i.e., with 70 to 100 multicast groups) that DST-LU suffers from congestion on few Critical Links. At heavy traffic workloads (i.e., with more than 100 multicast groups) L2BM suffers from more congestion with 5% more critical links compared to DST-PL and DST-LU as shown in Figure 6c. In return, L2BM improves the bandwidth demands acceptance ratio by 5% and 15% compared to DST-LU and DST-PL, respectively. Overall, for a given reserved link capacity for guaranteed-bandwidth multicast traffic, L2BM is able to serve a higher percentage of bandwidth demand multicast requests compared to DST-LU and DST-PL.



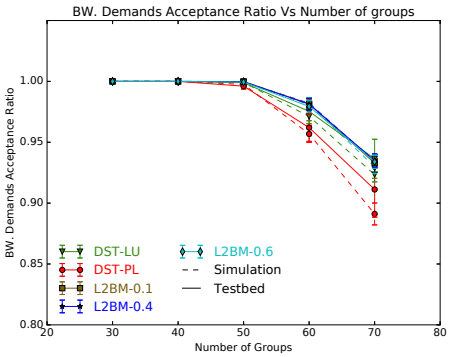
(a) Avg. Link Util.



(b) StdDev. Link Util.

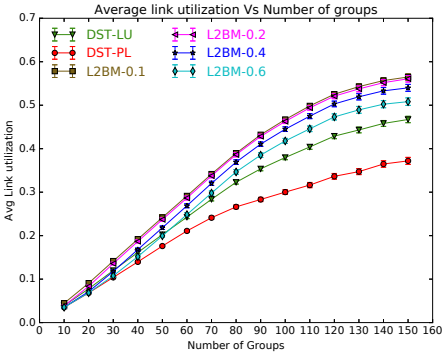


(c) Percentage of Critical Links

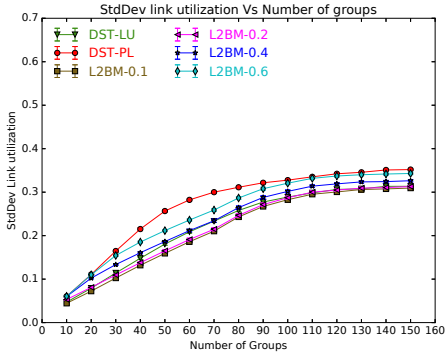


(d) Bandwidth Demands Acceptance Ratio

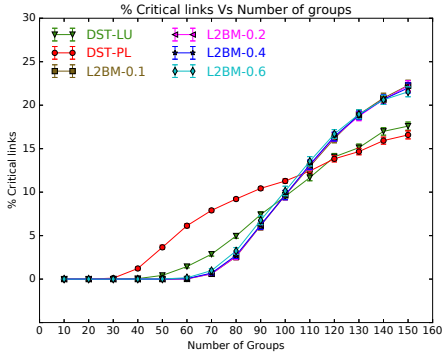
Fig. 5: Comparison of Testbed and Simulation results



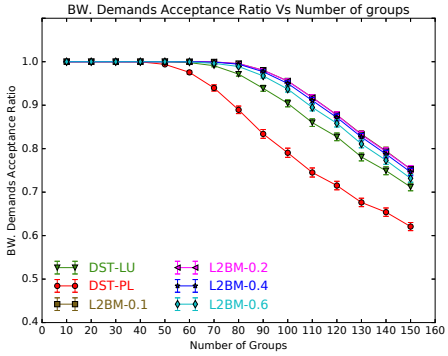
(a) Avg. Link Util.



(b) StdDev. Link Util.



(c) Percentage of Critical Links



(d) Bandwidth Demands Acceptance Ratio

Fig. 6: Results using Concentrated workloads without churn

3) Validation of the $M/M/\infty$ Model using Workloads with Churn: Next, we validate the $M/M/\infty$ queue based on the simulation model for the scenario with churn and the above mentioned link utilization metrics. We take an example run of moderate traffic workload of 130 multicast groups with Concentrated traffic scenario to assess the performance of the algorithms over the run duration to study the traffic pattern generated by the model over the run. Figures 7a, 7b and 7c show the Exponential Moving Average of the Avg and StdDev measures and the Critical Links metric at every second of the workload run. As we can observe, after continuous initial increase till 150 seconds in the value of all the metrics, the effect of churn begins and the links' utilization metrics maintain the value of all the metrics in a small range, e.g., the Avg link utilization value ranges from 0.3 to 0.4 for DST-PL, 0.4 to 0.5 for DST-LU and 0.6 to 0.6 for L2BMs. This can be explained by a first period in which multicast receivers join, followed by a period where receivers both join and leave based on $M/M/\infty$ queue model. Concerning the StdDev and the Critical Links metrics in Figures 7b and 7c, the performance of the algorithms are indistinguishable during the execution period. Therefore, we execute 500 runs of the workload and study the bandwidth demands acceptance ratio metric as shown in Figure 7d. The x-axis shows different values (1, 2, 5, 9, 12, 25, 50 and 80 Mbps) of bandwidth demands corresponding to multicast join requests whereas the y-axis shows the acceptance ratio obtained for each demand. For this workload, L2BM-0.1 accepts 2-3% more requests than DST-LU whatever bandwidth demands values of join requests. We note that none of the algorithms are heavily biased towards some bandwidth demands values of join requests. For all the algorithms, the join request acceptance ratio gradually drops when increasing the values of the bandwidth demands. The main reason is that low bandwidth demands values can be satisfied using small residual capacity available on heavily utilized links.

4) Simulation Results using Workloads with Churn: We further evaluate the algorithms using workloads with churn and different number of multicast groups varying from 10 to 150. Figures 8a and 8b show the Avg and StdDev measures of links' utilization. L2BM-0.1 uses 5% and 15% more Avg link utilization compared to DST-LU and DST-PL, respectively. see Figure 8a. Similar to the scenario without churn, all the algorithms accept 100% of guaranteed-bandwidth multicast requests for the low traffic workloads with 10 to 60 multicast groups, see Figure 8d. However, for the same workloads, L2BM does not use any link above 90% of the capacity as shown in Figure 8c. There is no significant difference between DST-LU and L2BM for bandwidth acceptance ratio in moderate traffic workloads(i.e., 70 to 100 multicast groups), but DST-LU suffers from congestion on 3-4% more links as shown in Figure 8c. As we increase the workload to heavy traffic, L2BM increases the percentage of Critical Links by 2-3%, but it increases the *Bandwidth Demand Acceptance Ratio* by 8-10% and 3-4% compared to DST-PL and DST-LU, respectively for $\theta_{init} = 0.1$.

Overall, in presence of low concentrated traffic, L2BM increases the Avg link utilization (due to the early load balancing that results in using paths longer than the shortest one) but minimizes the traffic concentration on a set of links near hot spots. Thereby, it provides more residual bandwidth on these links from θ_{max} link capacity allocated for guaranteed-bandwidth multicast traffic, being more accommodating and friendly to the best-effort traffic on these links. When the traffic is highly concentrated, L2BM is able to accept a higher number of guaranteed-bandwidth multicast requests than the other algorithms by using the threshold based technique. By doing so, L2BM allows to accept more join requests at the cost of slightly increasing the number of Critical Links as shown in 6c. Hence, it is able to more efficiently utilize the allocated network bandwidth on the links.

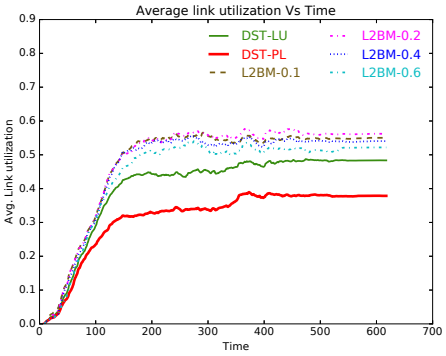
VI. Related Work

Several approaches have been proposed to provide multicast solutions that leverage the logically centralized SDN control plane [2], [3], [4], [6], [5], [7], [8]. However, they do not address the specific scalability issue emerging from centralized processing of group membership messages at the controllers and do not provide low-cost best-effort friendly traffic engineering policy in presence of QoS guaranteed multicast traffic.

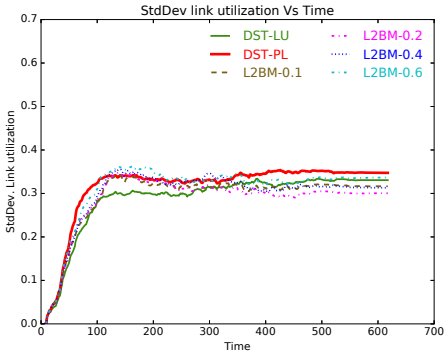
On the Scalability of the SDN Control Plane and of Multicast Group Management

First, we study the approaches addressing multicast group membership management regarding the SDN centralized control plane. In [2], authors have proposed an SDN-based multicasting scheme to implement XMPP chat sessions, referred as P2PChat. They propose to run a P2PChat Server to aggregate XMPP subscription messages, assign multicast IP addresses and submit IPs of the subscribers to a controller. However, this approach is not scalable because every join/leave request from the subscribers will result in an update message from the P2PChat Server to the controller. Also, the work does not address scenarios involving inter-domain multicast and group membership management in ISP networks, unlike us. CastFlow [3] proposes a clean-slate approach for multicast with membership churn, which does not use IGMP. A server component named Topology Server is responsible for multicast group configuration, and handles group join and leave events, but the replacement mechanism for IGMP messages is not detailed. Also, CastFlow does not address the scalability issue arising from centralized processing of group membership messages from receivers in real-world multicast scenarios. MultiFlow [4] is another clean-slate approach that uses IGMP messages for group membership management. The work focuses on decreasing the delay in the configuration of the multicast groups. In [6], some optimization is proposed to prevent IGMP flooding in the network domain of the controller but still, all the IGMP membership messages have to reach the controller, which may overwhelm it for large multicast sessions.

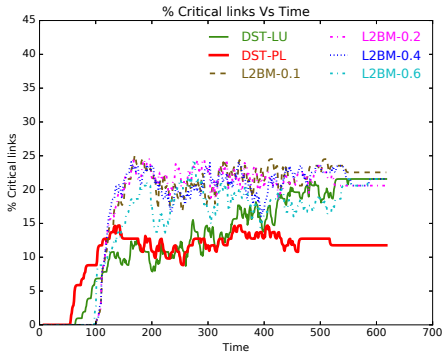
In [8], authors have proposed the Software Defined Multicast (SDM) service and some APIs for efficient delivery of OTT multicast traffic in ISP networks. However, *SDM* relies on a centralized group management mechanism for its



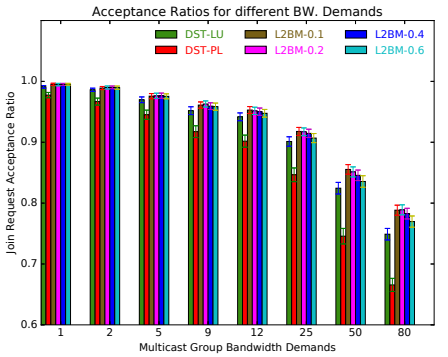
(a) Avg. Link Util vs. Time



(b) StdDev. Link Util vs. Time

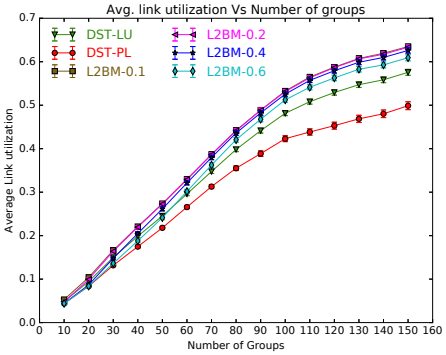


(c) Percentage of Critical Links

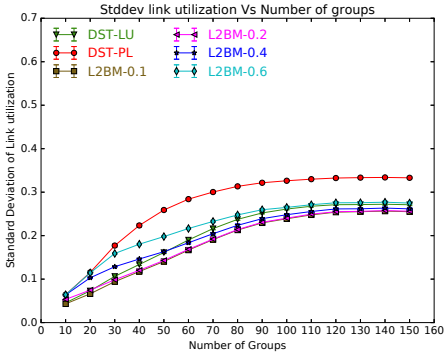


(d) Join Acceptance Ratio

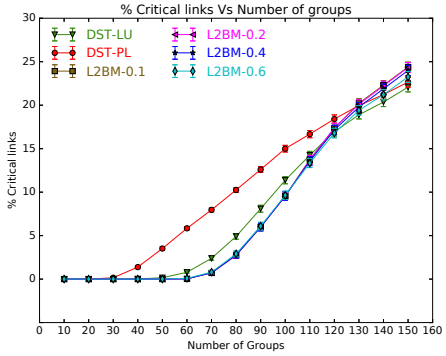
Fig. 7: Results using Concentrated workloads with churn and 130 multicast groups



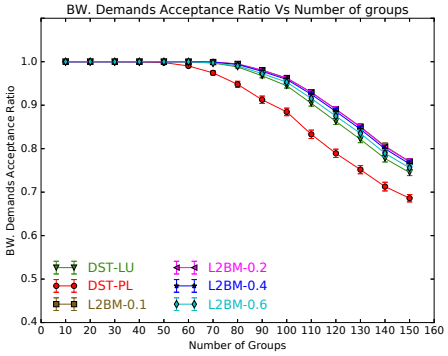
(a) Avg. Link Util.



(b) StdDev. Link Util.



(c) Percentage of Critical Links



(d) Bandwidth Demands Acceptance Ratio

Fig. 8: Results of L2BM, DST-PL and DST-LU using Concentrated workloads with churn

implementation in ISP networks and group management events are received from OTT content providers via some APIs, instead of using IGMP messages. Transparent to OTT clients, the *SDM* service converts OTT unicast traffic to multicast and vice versa. Multicast-to-unicast translation at the edge switches in the network may be an interesting approach for small groups but it is not able to scale with very large groups.

Regardless multicast, distributed [14], [15], [16] and hierarchical [18], [19] approaches have been proposed in the literature to increase the scalability of the SDN control plane. However, sharing the network view and applications state across multiple controller instances is complex, costly and can lead to unacceptable latency to handle data plane events. Dynamically increasing the number of controllers (ElastiCon [17]) and redistributing switches across them can help to handle surge in control workload (i.e., IGMP join/leave requests) [17], but control elements still have to receive and handle all the IGMP messages and to coordinate to ensure a consistent view, which is problematic for large-scale multicast. Concerning hierarchical approaches, they require southbound protocol specific mechanisms to maintain consistency across the controllers within different hierarchies, which decreases the visibility of the global network. Our approach exploits the hierarchy of the network but does not use specific control messages or event type of southbound protocols as it is the case in Kandoo [18] to communicate between controllers at different levels in the hierarchy. Instead, messages exchanges among MNFs are done independently of the southbound protocol. This gives flexibility in programming network functions with the required degree of distributed processing and state management without altering the southbound control protocols neither the controllers.

Recently, adding stateful programmable control logic in the switches has been proposed to offload logically centralized controllers in OpenState [20], POF [21], P4 [43] and SNAP [44]. With the availability of Protocol Independent Switch Architecture [45], it might be possible to provide the MNFs functionality by programming the switches. As switch technology evolves and provides stateful programmable control logic, it would be interesting to explore the possibility of implementing MNFs using such approaches.

In [7], a network architecture similar than ours is proposed, where the SDN controller is responsible for setting up routing paths and NFV nodes to run specific NFs like packet filtering and video transcoding. However, they do not tackle the group membership scalability issue and their multicast routing algorithm targets a different objective than ours, minimizing the sum of network link and node utilization.

On Multicast Routing with Guaranteed-Bandwidth

Multicast routing with QoS guarantee and traffic engineering has a rich literature. However, due to limited deployment of multicast and distributed architecture of ISP networks, the bulk of the proposed algorithms e.g., [9], [10], [11], [12], [13] have never been used in practice by multicast routing protocols like DVMPR, PIM, and MOSPF. Note that none of these algorithms considers the impact of guaranteed-bandwidth multicast flows on best-effort or other lower priority guaranteed-QoS traffic.

In [9], authors have defined the Maximum Multicast Flow (MMF) problem. They propose to update the link weights and to use a greedy heuristic of nearest neighbor for the Dynamic Steiner Tree. In contrast, L2BM does not make such assumption neither requires prior knowledge of future demands. Also, the computational time complexity of our algorithm is equivalent to the one of breadth-first search, while to update the link weights, MMF has $O(|\mathcal{E}|^2 * \log^2 Max(C_{uv}))$ time complexity.

Multiple QoS constrained Dynamic Multicast Routing (MQ-MDR) [10] is a mechanism proposed to guarantee jitter, delay and loss metrics apart from bandwidth and to minimize the traffic load in the network. However, unlike L2BM, MQ-MDR requires that multicast join requests are tagged with some participation duration information, which are used to assign weights to the links.

Other objectives like minimizing the maximum link utilization can be used to efficiently serve future multicast join requests. For instance, the Hop-count Constrained Multicast Tree heuristic [11] aims to find a path connecting a receiver to the existing multicast tree that minimizes the maximum link utilization of the path and with a shorter length than with the Hop-count constraint. The Multi-objective Multicast routing Algorithm (MMA) [12] considers the tree cost as another metric to minimize along with maximum link utilization. However, decreasing the Avg link utilization does not guarantee a higher acceptance ratio of guaranteed-bandwidth join requests as we show in Section V-E. L2BM attempts both to reduce the Avg link utilization (to be friendly with best-effort traffic) and to accept more guaranteed-bandwidth join requests.

In [13], the Nearest Node First Dijkstra Algorithm (NNFDA) is proposed, corresponding to DST-PL described in Section V-A. Our results show that L2BM is able to perform better than DST-PL both in terms of load-balancing and bandwidth demands acceptance ratio. An alternative load-balancing solution for multicast traffic consists in splitting each multicast flow in multiple thin-streams sent in different multicast trees, as proposed in DYNSDM [8]. However such an approach requires packets reordering at the receivers of the multicast stream, which increases jitter.

VII. Conclusion

In this paper we propose an NFV-based approach to overcome the multicast scalability issue of centralized SDN architectures. Then we present a novel threshold-based load balancing algorithm to deploy at low cost a guaranteed-bandwidth multicast service that nicely cohabits with best effort traffic. Our solution uses a traffic engineering mechanism to split the network bandwidth between best-effort traffic and guaranteed-bandwidth multicast traffic. We show that it is able to accept 5% more bandwidth demands compared to state-of-the-art algorithms for traffic scenarios representing flash crowd and prime-time streaming of videos. The source code and scripts used to evaluate our solution are made available to the community to ease reproduction of the experimental

results[8]. In the future, we plan to explore how to achieve greater programmability for in-network state management and computation at the edge of ISP networks.

ACKNOWLEDGEMENTS

This work has been partly supported by Inria and the Japanese Society for the Promotion of Science (JSPS) in the context of the UHD-on-5G associated team and by the French ANR under the #ANR-13-INFR-013 DISCO project and the "Investments for the Future" Program reference #ANR-11-LABX-0031-01. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

REFERENCES

[1] C. Diot, B. Levine *et al.*, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 14, no. 1, pp. 78–88, Jan 2000.
[2] K.-K. Yap, T.-Y. Huang, B. Dodson, M. S. Lam, and N. McKeown, "Towards Software-friendly Networks," in *ACM APSys*, 2010.
[3] C. A. Marcondes *et al.*, "Castflow: Clean-slate multicast approach using in-advance path processing in programmable networks," in *IEEE ISCC*, 2012.
[4] L. Bondan, L. F. Müller, and M. Kist, "Multiflow: Multicast clean-slate with anticipated route calculation on openflow programmable networks," *Journal of Applied Computing Research*, vol. 2, no. 2, pp. 68–74, 2013.
[5] S. Tang, B. Hua, and D. Wang, "Realizing video streaming multicast over SDN networks," in *EAI CHINACOM*, Aug 2014, pp. 90–95.
[6] A. Craig, Nandy *et al.*, "Load balancing for multicast traffic in SDN using real-time link cost modification," in *IEEE ICC*, June 2015.
[7] S. Q. Zhang *et al.*, "Routing Algorithms for NFV Enabled Multicast Topology on SDN," *IEEE TNSM*, vol. 12, no. 4, Dec 2015.
[8] J. Ruckert *et al.*, "Flexible, Efficient, and Scalable Software-Defined OTT Multicast for ISP Environments with DynSDM," *IEEE TNSM*, 2016.
[9] M. Kodialam, T. Lakshman, and S. Sengupta, "Online multicast routing with bandwidth guarantees: a new approach using multicast network flow," *IEEE/ACM ToN*, vol. 11, no. 4, pp. 676–686, 2003.
[10] D. Chakraborty *et al.*, "A dynamic multicast routing satisfying multiple QoS constraints," *Wiley IJNM*, vol. 13, no. 5, 2003.
[11] Y. Seok, Y. Lee, Y. Choi, and C. Kim, "Explicit multicast routing algorithms for constrained traffic engineering," in *IEEE ISCC*, 2002.
[12] J. Crichigno and B. Baran, "Multiobjective multicast routing algorithm for traffic engineering," in *IEEE ICCCN*, Oct 2004, pp. 301–306.
[13] S. Youm, T. Shon, and E.-J. Kim, "Integrated multicast routing algorithms considering traffic engineering for broadband iptv services," *EURASIP JWCN*, vol. 2013, no. 1, p. 127, 2013.
[14] S. H. Yeganeh and Y. Ganjali, "Beehive: Simple Distributed Programming in Software-Defined Networks," in *ACM SOSR*, 2016.
[15] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *IEEE NOMS*, May 2014, pp. 1–4.
[16] T. Koponen, M. Casado *et al.*, "Onix: A Distributed Control Platform for Large-scale Production Networks," in *ACM OSDI*, vol. 10, 2010.
[17] A. A. Dixit, F. Hao, S. Mukherjee *et al.*, "ElastiCon: An Elastic Distributed SDN Controller," in *ACM/IEEE ANCS*, 2014.
[18] S. H. Yeganeh and Y. Ganjali, "Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications," in *ACM HotSDN*, 2012.
[19] Y. Fu *et al.*, "A Hybrid Hierarchical Control Plane for Flow-Based Large-Scale SDNs," *IEEE TNSM*, vol. 12, no. 2, June 2015.
[20] G. Bianchi *et al.*, "OpenState: Programming Platform-independent Stateful OF Applications Inside the Switch," *ACM CCR*, vol. 44, no. 2, 2014.
[21] H. Song, "Protocol-oblivious Forwarding: Unleash the Power of SDN Through a Future-proof Forwarding Plane," in *ACM HotSDN*, 2013.
[22] M. Santos, B. Nunes, K. Obraczka, T. Turletti, B. de Oliveira, and C. Margi, "Decentralizing SDN's Control Plane," in *IEEE LCN*, 2014.
[23] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3," RFC 3376, 2015.
[24] B. Pfaff, J. Pettit, T. Koponen *et al.*, "The Design and Implementation of Open vSwitch," in *12th ACM NSDI*, May 2015.
[25] H. Soni, W. Dabbous, T. Turletti, and H. Asaeda, "Scalable Guaranteed-Bandwidth Multicast Service in Software Defined ISP networks," in *IEEE ICC*, May 2017.
[26] B. Fenner, M. Handley, I. Kouvelas, and H. Holbrook, "PIM-SM: Protocol Specification (revised)," RFC 7761, Mar. 2016.
[27] "Network Functions Virtualisation (NFV). ETSI Industry Specification Group (ISG)," http://portal.etsi.org/NFV/NFV_White_Paper3.pdf.
[28] "HTB - Hierarchy Token Bucket, ," https://linux.die.net/man/8/tc-htb.
[29] N. McKeown, T. Anderson *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, 2008.
[30] J. Evans, A. Begen, J. Greengrass, and C. Filsfils, "Toward lossless video transport," *IEEE Internet Computing*, vol. 15, pp. 48–57, 2011.
[31] "LINC Switch," https://github.com/FlowForwarding/LINC-Switch.
[32] "CPqD Software Switch," https://github.com/CPqD/ofsoftswitch13.
[33] "HTB - Hierarchy Token Bucket, ," http://www.pica8.com/resources/technology.
[34] "6WIND Debuts 195 Gbps Accelerated Virtual Switch," http://www.6wind.com/blog/.
[35] B. Pfaff and B. Davie, "The Open vSwitch Database Management Protocol," RFC 7047, Dec. 2013.
[36] "libfluid. The ONF OpenFlow Driver," http://opennetworkingfoundation.github.io/libfluid/.
[37] B. M. Waxman, "Routing of multipoint connections," *IEEE JSAC*, vol. 6, no. 9, pp. 1617–1622, Dec 1988.
[38] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-defined Networks," in *ACM HotSDN*, 2010.
[39] H. Soni, D. Saucez, and T. Turletti, "DiG: Data-centers in the Grid," in *IEEE NFV-SDN*, Nov 2015, pp. 4–6.
[40] "Internet2 AL2S Topology," https://noc.net.internet2.edu/i2network/advanced-layer-2-service/maps-documentation/al2s-topology.html.
[41] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin, "A hierarchical characterization of a live streaming media workload," in *ACM IMW*, Nov 2002, pp. 117–130.
[42] N. Hagiya *et al.*, "Concentrated traffic in non-hierarchical networks under alternate routing scheme: a behavior analysis," in *IEEE GLOBECOM*, 1993.
[43] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM CCR*, vol. 44, no. 3, Jul. 2014.
[44] M. T. Arashloo *et al.*, "Snap: Stateful network-wide abstractions for packet processing," in *ACM SIGCOMM*, New York, NY, USA, 2016.
[45] P. Bosshart *et al.*, "Forwarding Metamorphosis: Fast Programmable Match-action Processing in Hardware for SDN," in *ACM SIGCOMM*, New York, NY, USA, 2013.

[8] See URL https://team.inria.fr/diana/software/l2bm/